mapping a path of control flow on the stack from any start point in a selected method to the destination program counter; and

simulating stack actions for executing bytecodes along said path,

wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and

recording unprocessed targets in a pre-allocated memory location from any branches in the first linear bytecode sequence for future processing, and

where the destination program counter was not reached during an earlier processing of a linear bytecode sequence,

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets in said pre-allocated memory location from any branches in the additional linear bytecode sequence for future processing,

wherein said processing of said bytcode sequences is performed without modifying the program bytecodes in the stack.

REMARKS

Reconsideration of the above-identified application, in view of the following remarks, is respectfully requested.

In the FINAL REJECTION dated February 24, 2003, the Examiner again rejected Claims 1-3,10-13, and 20-23 under 35 U.S.C. §102(e) as allegedly being unpatentable over Agesen (U.S. Patent No. 6,047,125 hereinafter "Agesen1"). The Examiner further rejected Claims 4-5 and 14-15 under 35 U.S.C. §103(a) as allegedly being unpatentable over

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

Agesen1 in view of Agesen (U.S. Patent No. 5,909,579 hereinafter "Agesen2"). The Examiner then rejected Claims 6-7, 9, 16-17 and 19 under 35 U.S.C. §103(a) as allegedly being unpatentable over Agesen1 in view of Gosling (U.S. Patent No. 5,668,999). The Examiner further rejected Claims 8 and 18 under 35 U.S.C. §103(a) as allegedly being unpatentable over Agesen1 in view of Gosling and further in view of O'Connor (U.S. Patent No. 6,098,089).

More particularly, the Examiner has maintained his rejection of claims 1-3, 10-13, and 20-23 on the same grounds as in the first rejection as allegedly being anticipated by Agesen1.

Applicants respectfully disagree. As explained in the present specification at page 7, lines 3-16, the present invention is directed to:

> ...a method for mapping a valid stack up to a destination program counter through mapping a path of control flow on the stack from any start point in a selected method to the destination program counter and simulating stack actions for executing bytecodes along said path. In order to map a path of control flow on the stack, bytecode sequences are processed linearly until the control flow is interrupted. As each bytecode sequence is processed, unprocessed targets from any branches in the sequence are recorded for future processing. The processing is repeatedly iteratively, starting from the beginning of the method and then from each branch target until the destination program counter has been processed."

As part of the bytecode sequence processing a "virtual stack" is constructed (See, step 114, Figure 1A of the present specification) in a pre-allocated memory area (which may comprise a portion of the stack, See Figure 5, step 508). That is, as argued previously in applicants prior response of December 13, 2003, the present invention deals with the dynamic nature of the stack by leaving open a small area of memory in the stack and tagging it for dynamic

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

mapping. This, it is respectfully submitted, is neither taught nor suggested in the prior art and, in light of this, Claim 1 is being amended herein to clarify the fact that in the present invention, i.e., in result of simulating stack actions, a virtual stack is constructed for storage in an pre-allocated memory location which area has been tagged for dynamic mapping. See present specification at page 18, lines 16-20.

Additionally, as applicants have previously argued, the substance of the system described in Agesen1 is that it provides for modifying the program bytecodes in the stack, see Agesen1 at col.5, lines 23-25, for example. That is, every embodiment described in Agesen1 includes the step of rewriting some code in the program. The present invention, on the other hand, does not define rewriting any code in the program either in the description or in the claims. Consequently, Claim 1 is being further amended herein to clarify the fact that in the present invention, the processing of the bytcode sequence is performed without modifying the program bytecodes in the stack, i.e., no rewriting of code is performed at all.

As these features were described and argued in applicants' prior response, but not set forth in the claims, applicants now respectfully amend the claims for clarification purposes to set forth this subject matter as was relied upon by the applicants in their arguments submitted with their prior response of December 13, 2002. Respectfully, no new matter is being entered by these amendments. As Agesen1 does not teach or suggest any of these steps at all, as such, there can be no anticipation.

Moreover, in the Office Action, the examiner (in paragraph 2) has rejected Claims 1 and 11 and states that Agesen1 teaches simulating stack actions for executing bytecodes, e.g., at col.3, lines 62-65 and col.11, lines 59-63. It is respectfully submitted that, upon a careful reading of Agesen1, there is nothing approaching simulating stack actions at

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

the cited passage or any other passage. Agesen1 deals with instructions in code being processed, as exemplified in the cited passage at col.5, l.32, where it states: "during execution of the instructions". The passage at col.3, lines 62-65 is not a statement of any operation performed by Agesen1, but merely an example of a bytecode (e.g., `iadd`) that is found in the target program. As to "execution" at col.11, lines 59-63, the code in Figure 7 is also an example of what bytecodes the invention might encounter in code created by a Java compiler - see the general description of Figure 7 at col. 10, lines 49-52 in Agesen1. Further, described at col.11, lines 64-65 in Agesen1, are the conflicts that the Agesen1 invention is intended to resolve.

The examiner states that Agesen1 describes "mapping a path of control flow on the stack from any start point in a selected method to the destination program counter by locating a linear path from the beginning of the method to the destination program counter and iteratively processing a bytecode sequence for each branch until said destination program counter is reached." In support of this contention, the examiner states that at col.7, lines1-5, Agesen1 uses a stack map to map controls to a program counter. However, applicants submit that the sentence immediately preceding the cited passage is critical to the passage, the whole of which reads:

> "The conflicts are caused by certain conditions set by rules and exceptions dictated by Java's bytecode verifier. They made it impossible for a collector, which relies on only the program counter to determine the stack map to use, to determine conclusively whether a local variable with an assigned stack frame slot represents a reference. Such conflicts are eliminated by rewriting selected bytecodes to refer to a different variable or adding additional bytecodes to initialize the variable."

Thus, respectfully, the applicants disagree with the Examiner's contention and submit that the

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

complete passage describes a condition in the Java code that the invention of Agesen1 is intended to resolve. Neither the cited passage nor any other passage in Agesen1 describes locating a linear path from the beginning of a method to a destination program counter to map a path of control flow. Therefore, applicants' respectfully submit that Agesen1 discloses neither of the steps in any of claims 1-3, 10-13 and 20-23 nor any similar step, and that Agesen1 thus does not anticipate the invention claimed therein. The applicant respectfully requests withdrawal of the rejected claims under 35 U.S.C. §102(e).

At paragraph 6 of the Office Action, the Examiner further stated that the features not shown in the reference, and relied upon by the applicants, are not recited in the claims. The applicant submits with respect that the particular features are discussed to show that the reference is directed to resolving conflicts indicating an undeterminable variable type in garbage collection, and is not relevant to the present invention which is directed to mapping the shape of the stack. The present invention is not a garbage collection method, although it may be used prior to garbage collection. In any event, the applicants submit that, in view of the discussion and amendments to Claims 1 and 11 above, the above point is now moot.

In the Office Action, the Examiner further rejected Claims 2, 3, 12 and 13 in similar fashion, alleging that Agesen1 teaches processing a first linear bytecode sequence and an additional one until the control flow is interrupted. The reference to col.5, lines 19-27, shows an example of a conflict that Agesen1 attempts to resolve, and not how Agesen1 resolves it. As to the cited passage at col.7, lines 42-44, the interrupt controller, 135 in Agesent1, Figure 3, is merely a component of a standard well-known computer system. It is not shown in Agesen1 that the interrupt controller performs a particular step similar to steps in claims 2, 3, 12 or 13.

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

As to the passage at col.11, lines 54-59, cited by the Examiner, the applicant submits that while an exception handler as described by Agesen1 may in some situations interrupt control flow, Agesen1 does not show that any unprocessed targets are recorded for future processing. The only item recorded is the reference for the exception that caused entry into the exception handler, which is quite different from the virtual stack construction and storage step of the amended Claims 1 and 11.

The Examiner further states that "Ageson [sic] inherently teaches: the destination program counter was not reached during an earlier processing of a linear bytecode sequence [and] the information stored in the reference can be used at a future time for processing." Respectfully, the information stored in the reference is not information about an unprocessed target. Rather it is about an exception that caused entry into the handler. Applicants submit that the cited passage does not teach or suggest a step similar to the steps of Claims 2, 3, 12 or 13, and that the rejection based thereon should be withdrawn. Further, claims 2, 3, 12 and 13 depend from Claims 1 and 11 respectively, which the applicant has already shown to be patentable. Thus the applicant submits that the rejection of claims 2, 3, 12 and 13 under 35 U.S.C. §102(e) should be withdrawn.

Further in the Office Action, as the Examiner has rejected Claim 23 for the same reasons as Claims 1-3. Claim 23, already includes the limitations of Claims 1-3 and additionally has been amended herein to further set forth the limitations added to amended Claim 1. Nonetheless, it has been shown that Claims 1-3 are allowable, and consequently, applicants' respectfully request the Examiner withdrawal of should be able to use the same arguments to overcome this rejection.

With respect to the Examiner's rejection of Claims 4-5 and 14-15 under 35

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

U.S.C. §103(a) as allegedly being unpatentable over Agesen1 in view of Agesen2, Agesen2 describes, as stated in the summary of the invention at col. 3, line 63 to col.4, line 17, a method in which:

> "...live pointer information for a stream of bytecodes is precomputed for each bytecode. The precomputed full live pointer information is stored only for bytecodes at predetermined intervals in the stream. Between the bytecodes for which full live pointer information is stored, changes in the live pointer information produced by each bytecode are encoded using a suitable compressive coding and stored. Later, when garbage collection is initiated, the full live pointer information for the nearest bytecode preceding the desired bytecode boundary is retrieved along with the intervening coded changes. The changes are decoded and applied to the retrieved live pointer information to generate the live pointer information at the desired bytecode boundary. In accordance with one embodiment of the invention, the live pointer changes are delta encoded so that each code contains information relating to the live pointer changes produced by a bytecode from the live pointer information as modified by the previous delta code. In accordance with another embodiment of the invention, the delta coded changes are encoded with a Huffman encoding scheme.

The basic invention in Agesen2 is choosing to store full information for only every *n*th bytecode in the stream, and recreating the deltas in between them. The Examiner states that Agesen2 "teaches a 'bytecode analyzer mechanism' which determines the changes which the bytecode makes to the live pointer locations (col.8, lines 20-24)." He further states that "the system has breakpoints or computation stops at bytecode boundaries for determining live pointer information (col.3, lines 10-13)."

The applicants respectfully submit that Agesen1 and Agesen2 combined do not disclose either mapping a path of control flow or simulating stack actions, as discussed above with respect to claims 1 and 11, from which claims 4, 5, 14 and 15 depend. Neither do they disclose any steps that are similar to, or that suggest the use of, those two elements. Thus since the two references do not contain or suggest all the elements of the claims, the

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

applicants submit respectfully that the invention of Claims 4, 5, 14 and 15 is not obvious by the combined references and the Examiner is respectfully requested to withdraw the rejections of Claims 4-5 and 14-15 under 35 U.S.C. §103(a).

With respect to the rejection of Claims 6-7, 9, 16-17 and 19 under 35 U.S.C. §103(a) as allegedly being unpatentable over Agesen1 in view of Gosling, the Examiner states that Gosling supplies, col.5, lines 21-29, the step of generating a virtual stack that is omitted from Agesen1. However, Claims 1 and 11, from which Claims 6 and 16 respectively depend, are restricted to mapping a path of control flow only from an arbitrary start point up to a destination program counter. Therefore the virtual stack thus created is only a stack for part of the program, unlike Gosling which creates a virtual stack for the entire program, i.e., "used in the same way as a regular stack..", (see col.5, lines 35-40 of Gosling). That is, the method of the present invention eliminates the time and cost of generating a complete virtual stack in the manner of Gosling. Further, as the limitations of amended Claims 1 and 11 have been discussed above, and their distinctions over the cited prior art have been pointed out, it is submitted that the combined references of Agesen1 and Gosling do not include any similar steps that would suggest all the steps of Claims 6 and 16, which depending from amended Claims 1 and 11, respectively. The applicant respectfully submits that Claims 6 and 16 have not been shown to be obvious over the cited art and respectfully requests that the rejection of Claims 6 and 16 under 35 U.S.C. §103(a) be withdrawn.

Likewise, regarding Claims 7 and 17, these each depend respectively from Claims 6 and 16, and include the limitations of the parent claims as discussed above. It is respectfully submitted that Claims 7 and 17 are thus allowable also. The applicant respectfully requests that the rejection under 35 U.S.C. §103(a) be withdrawn.

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

Regarding the rejection of Claims 9 and 19, depending from claims 7 and 17 respectively, the storing of a bitstring to a pre-allocated area in the memory (stack) is described at page 19, lines 1-4 of the disclosure. Although storing information to a pre-allocated area is known in general, the claimed pre-allocated area is used when a special event occurs. In contrast, Agesen1 stores the value at a position selected not by the invention but by the compiler, to wit:

> "The compiler selected variable 3 to store the value for itmp so bytecode 704 stores the value (1) from the top of the operand stack in variable 3."

This storage is simply matching the storage location to the location chosen by the compiler. Further, the applicant respectfully submits that the combination of Agesen1 and Gosling does not include nor suggest all of the elements of Claims 9 and 19, and requests that the rejection under 35 U.S.C. §103(a) be withdrawn.

The Examiner further rejected Claims 8 and 18 under 35 U.S.C. §103(a) as allegedly being unpatentable over Agesen1 in view of Gosling and further in view of O'Connor. Particularly, the examiner states that although neither Agesen1 nor Gosling teaches storing the bitstring on a heap, O'Connor states that it is well known in the art that "garbage collection" of "heap-allocated storage" is an "attractive model for dynamic memory management", See Col.1, lines 39-42 of O'Connor. Claims 8 and 18 depend from claims 7 and 17 respectively, whose allowability has been discussed above. Claims 8 and 18 add the limitation that the step of storing the bitstring comprises storing the bitstring to the selected method as compiled on a heap. The cited passage from O'Connor does not indicate anything about the stack map being placed in the heap within the compiled method. In general, O'Connor discusses ordinary stores to the heap but not the storage of a stack map. Furthermore, it has been discussed above that Agesen1 does not include nor suggest the steps

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

of the claims from which claims 8 and 18 depend. Therefore, it is respectfully submitted that the combined references do not include all the limitations of Claims 8 and 18, and therefore the applicant respectfully requests the withdrawal of the rejection of Claims 8 and 18 under 35 U.S.C. §103(a).

Attached hereto is a marked-up version of the changes made to the specification by the current amendment.

In view of the foregoing, Applicant believes that this application is in condition for allowance and Applicants henceforth respectfully solicit such allowance. If the Examiner believes a telephone conference might expedite the prosecution of this case, Applicants respectfully request the Examiner to call the undersigned, Applicants' attorney, at: (516) 742-4343.

Respectfully submitted,

Steven Fischman
Registration No. 34,594

SCULLY, SCOTT, MURPHY & PRESSER
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343
SF:gc

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

Serial No.: 09/329,558 **D cket:** CA919980012US1 (12463)

MARKED-UP VERSION OF THE CHANGES MADE

IN THE CLAIMS:

Please amend Claims 1, 11 and 23 as follows:

1. (Twice Amended) A method for mapping a valid stack up to a destination program counter, comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter by locating a linear path from the beginning of the method to the destination program counter and iteratively processing a bytecode sequence for each branch until said destination program counter is reached; and

simulating stack actions for executing bytecodes along said path, and constructing a virtual stack for storage in a pre-allocated memory location,

wherein said processing said bytcode sequence is performed without modifying the program bytecodes in the stack.

11. (Twice Amended) A method for mapping a Java bytecode stack up to a destination program counter comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter by locating a linear path from the beginning of the method to the destination program counter and iteratively processing a bytecode sequence at each branch until said destination counter is reached; and

simulating stack actions for executing bytecodes along said path, and constructing a virtual stack for storage in a pre-allocated memory location,

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]

wherein said processing said bytcode sequence is performed without modifying the program bytecodes in the stack.

23. (Amended) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for mapping a valid stack up to a destination program counter, said method steps comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter; and

simulating stack actions for executing bytecodes along said path,

wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and

recording unprocessed targets in a pre-allocated memory location from any branches in the first linear bytecode sequence for future processing, and

where the destination program counter was not reached during an earlier processing of a linear bytecode sequence,

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets in said pre-allocated memory location from any branches in the additional linear bytecode sequence for future processing,

wherein said processing of said bytcode sequences is performed without modifying the program bytecodes in the stack.

Received from < 3 > at 4/23/03 4:42:44 PM [Eastern Daylight Time]